MODE D'EMPLOI DES LM629

La librairie Lm629.h contient les principales fonctions permettant de commander les deux LM.
Je ne vais pas détailler ici le fonctionnement interne de ces fonctions car nous avons simplement appliqué les procédures que nous avons trouvées sur un très bon site (http://manubatbat.free.fr/robot/Doc/LM629/LM6293.HTM) dont nous avons remis ici quelques extraits.

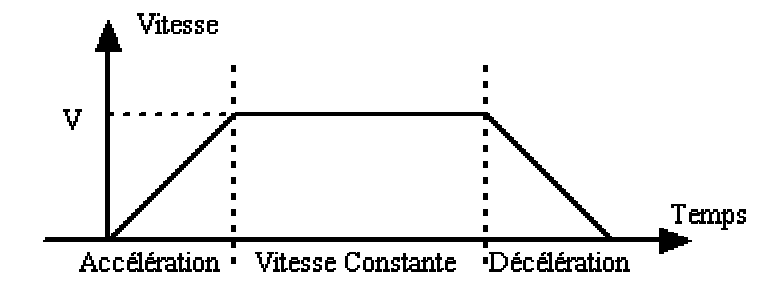
Nos fonctions portent les mêmes noms que sur le site. Pour nous faciliter la tache nous avons défini

- Une fonction char read (void) qui permet de lire un registre (en fonction de la commande que l'on a passé avant).
- Une fonction void write (void) qui permet d'écrire, une donnée ou une commande, dans un registre (en fonction de la commande que l'on a passé avant). Cette donnée ou commande est passée par le portb.

Ces deux fonctions se chargent de configurer le portb en entrée ou en sortie. Ce portb est connecté sur les deux LM à la fois, c'est pourquoi il est nécessaire d'activer le LM avec lequel vous voulez communiquer. Cela se fait par les bits LM1 et LM2 (pour le LM1 : LM1=0 et LM2=1 et pour activer LM2 : LM1=1 et LM2=0 ; ouais c'est en logique inverse, c'est un peu chiant, mais y suffit de faire gaffe).

<u>Comment faire pour charger une trajectoire ?</u>

Une trajectoire, c'est une accélération (qui fait aussi office de décélération), une vitesse maximum et une distance. Ces registres sont codés sur 32 bits, qu'il faut envoyer par 4 paquets d'octets (normal puisque la transmission ce fait en parallèle sur le portb).




```
void load_trajectory (char LM, char data1, char data2,
                      char a_h, char a_m_h, char a_m_l, char a_l,
                      char v_h, char v_m_h, char v_m_l, char v_l,
                      char p_h, char p_m_h, char p_m_l, char p_l)
```

LM : 1 pour LM1 et 2 pour LM2.
Data1 : cf. doc.

Data1 : cf. doc.
A_h : octet de poids le plus fort (high).
A_m_h : octet de poids moyen fort (middle high).
A_m_l : octet de poids moyen bas (middle low).
A_m_h : octet de poids le plus faible (low).

Idem pour v (vitesse) et pour p (position).

Lecture/Ecriture	Registre	Valeur
Lecture/Polling	Commande (PS=0)	Registre de status / Attente du bit **BUSY** (LM occupé) à 0
Ecriture	Commande (PS=0)	0x1F **(LTRJ)**
Lecture/Polling	Commande (PS=0)	Registre de status / Attente du bit **BUSY** (LM occupé) à 0
Ecriture	Donnée (PS=1)	**8 bits poids fort** bit 7 : Non utilisé. bit 6 : Non utilisé. bit 5 : Non utilisé. bit 4: Sens de rotation (*uniquement en mode vitesse*). bit 3: 1=**Mode Vitesse**, 0=**Mode Position**. bit 2: Arrêt doux du moteur. bit 1: Arrêt rapide du moteur. bit 0: Arrêt immédiat.
Ecriture	Donnée (PS=1)	**8 bits poids faible** bit 7 : Non utilisé. bit 6 : Non utilisé. bit 5 : Chargement de l'**accélération** *à suivre*. bit 4 : 1=Accélération relative, 0= Accélération absolue. bit 3 : Chargement de la **vitesse** *à suivre* . bit 2 : 1=Vitesse relative, 0=Vitesse absolue. bit 1 : Chargement de la **position à atteindre** *à suivre*. bit 0 : 1=Position relative, 0=Position absolue.
		Les 6 opérations suivantes ne sont effectuées que si le bit 5 était à 1
Lecture/Polling	Commande (PS=0)	Registre de status / Attente du bit **BUSY** (LM occupé) à 0
Ecriture	Donnée (PS=1)	**8 bits poids fort** de **l'accélération** -- en COUNT/SAMPLE/SAMPLE
Ecriture	Donnée (PS=1)	**8 bits poids mi-fort** de **l'accélération**
Lecture/Polling	Commande (PS=0)	Registre de status / Attente du bit **BUSY** (LM occupé) à 0

Ecriture	Donnée (PS=1)	**8 bits poids mi-faible** de **l'accélération**
Ecriture	Donnée (PS=1)	**8 bits poids faible** de **l'accélération**
		Les 6 opérations suivantes ne sont effectuées que si le bit 3 était à 1
Lecture/Polling	Commande (PS=0)	Registre de status / Attente du bit **BUSY** (LM occupé) à 0
Ecriture	Donnée (PS=1)	**8 bits poids fort** de **la vitesse** -- en COUNT/SAMPLE
Ecriture	Donnée (PS=1)	**8 bits poids mi-fort** de **la vitesse**
Lecture/Polling	Commande (PS=0)	Registre de status / Attente du bit **BUSY** (LM occupé) à 0
Ecriture	Donnée (PS=1)	**8 bits poids mi-faible** de **la vitesse**
Ecriture	Donnée (PS=1)	**8 bits poids faible** de **la vitesse**
		Les 6 opérations suivantes ne sont effectuées que si le bit 1 était à 1
Lecture/Polling	Commande (PS=0)	Registre de status / Attente du bit **BUSY** (LM occupé) à 0
Ecriture	Donnée (PS=1)	**8 bits poids fort** de **la position** -- en COUNT
Ecriture	Donnée (PS=1)	**8 bits poids mi-fort** de **la position**
Lecture/Polling	Commande (PS=0)	Registre de status / Attente du bit **BUSY** (LM occupé) à 0
Ecriture	Donnée (PS=1)	**8 bits poids mi-faible** de **la position**
Ecriture	Donnée (PS=1)	**8 bits poids faible** de **la position**

Comment sont codés ces registres ?

Donnée	Plage autorisée
Position	De **-2^30** à **2^30-1**, soit **0xC0000000** à **0c3FFFFFFF** ou encore de **1073741824** à **1073741823**
Vitesse	**Toute la plage. (0 à 65536,99998 soit 0x00000000 à 0xFFFFFFFF)**
Accélération	De **0** à **2^30-1**, soit de **0** à **0x3FFFFFFF** ou encore de **0** à **16383,99998**

Ainsi dans notre cas (codeur 500 impulsions par tour, réducteur de 6,25, quartz LM de 6MHz, diamètre roue de 8 cm et sachant que le LM 'voit' 4 fois plus d'impulsion qu'en réalité), on a :

$$position_{LM} = position_{m} * 50000$$

$$vitesse_{LM} = vitesse_{m/s} * 17$$

Exemples :

- Pour fixer une distance de 1 m, cela revient à envoyer au LM : 50000 soit :
  0, 0, 0xC3, 0x50 (4 paquets de 8 bits)

- Pour fixer une distance de -1 m, cela revient à envoyer au LM : 50000 en code complément à 2 soit :
  0xFF, 0xFF, 0x3C, 0xB0 (4 paquets de 8 bits)

- Pour fixer une vitesse de 0.5m/s, cela revient à envoyer au LM : 8,5 soit :
  0, 8, 0x80, 0 (4 paquets de 8 bits)

Pour l'accélération, soit vous n'y toucher pas (elle semble bien réglée) soit vous procédez à tâtons… Toutes les formules théoriques (pour la position, la vitesse et l'accélération) sont détaillées dans le rapport 2004 de l'UV PR sur la base roulante.

Exemple de fonctions à réaliser :

```
void avance (void)
{
   reset_int(1);
   reset_int(2);

   asm bcf LM1
   asm bsf LM2
   busy();

   load_trajectory (1, 00001000b, 00101000b,
                    0, 0, 2, 0,
                    0, 8, 0x80, 0,
                    0, 0, 0, 0);
```

```
    asm bsf LM1
    asm bcf LM2
    busy();

    load_trajectory (2, 00001000b, 00101000b,
                     0, 0, 2, 0,
                     0, 8, 0x80, 0,
                     0, 0, 0, 0);



    start_motion(1);
    start_motion(2);

}
```

Cette fonction permet d'avancer tout droit à 0.5m/s sans s'arrêter ; les LM sont placés en mode vitesse.

Pour comprendre le fonctionnement des LM regarder les fonctions et le site, ça devrait aller tout seul !

Comment charger les paramètres du filtres ?

Si vous ne charger aucun paramètre du PID, P=0 donc le robot n'avancera jamais !

Il vous suffit d'utiliser cette fonction :

```
void load_filter (char LM, char inter_deriv, char data,
            char p_h, char p_l, char i_h, char i_l,
            char d_h, char d_l, char limite_i_h, char limite_i_l);
```

LM : 1 ou 2
inter_deriv : 0 (pas important)
data : cf. site
char p_h, char p_l : parameter P (proportionnel) (en 16 bits)
char i_h, char i_l : idem pour I (intégral)
char d_h, char d_l : idem pour D (dérivé)

Comment sont codés ces registres ?

P, I et D sont des constantes, donc il n'y a pas de conversion à faire (hormis la décomposition en deux octets bien sur !).

Exemple :
Pour 10000 : 0x27, 0x10

Une fois que vous avez charger les paramètres du filtre, il ne sont pas pour autant appliqués !! Pour les appliquer, appelez la fonction void update_filter (char LM) ;

Load Filters Parameters :

Lecture/Ecriture	Registre	Valeur
Lecture/Polling	Commande (PS=0)	Registre de status / Attente du bit **BUSY** (LM occupé) à 0
Ecriture	Commande (PS=0)	0x1E (**LFIL**)
Lecture/Polling	Commande (PS=0)	Registre de status / Attente du bit **BUSY** (LM occupé) à 0
Ecriture	Donnée (PS=1)	**8 bits**, valeur de l'intervalle de dérivation. (cf tableau au dessus).
Ecriture	Donnée (PS=1)	**8 bits poids faible** indiquant ce qui sera chargé par la suite. bit 7 : Non utilisé. bit 6 : Non utilisé. bit 5 : Non utilisé. bit 4 : Non utilisé. bit 3 : Chargement du coefficient P *à suivre*. bit 2 : Chargement du coefficient I *à suivre*. bit 1 : Chargement du coefficient D *à suivre*. bit 0 : Chargement du terme limite d'intégration *à suivre* .
		Les 3 opérations suivantes ne sont effectuées que si le bit 3 était à 1
Lecture/Polling	Commande (PS=0)	Registre de status / Attente du bit **BUSY** (LM occupé) à 0
Ecriture	Donnée (PS=1)	**8 bits poids fort** de **P** -- en ???
Ecriture	Donnée (PS=1)	**8 bits poids faible** de **P**
		Les 3 opérations suivantes ne sont effectuées que si le bit 2 était à 1
Lecture/Polling	Commande (PS=0)	Registre de status / Attente du bit **BUSY** (LM occupé) à 0
Ecriture	Donnée (PS=1)	**8 bits poids fort** de **I** -- en ???
Ecriture	Donnée (PS=1)	**8 bits poids faible** de **I**
		Les 3 opérations suivantes ne sont effectuées que si le bit 1 était à 1
Lecture/Polling	Commande (PS=0)	Registre de status / Attente du bit **BUSY** (LM occupé) à 0
Ecriture	Donnée (PS=1)	**8 bits poids fort** de **D** -- en ???

Ecriture	Donnée (PS=1)	**8 bits poids faible** de **D**
		Les 3 opérations suivantes ne sont effectuées que si le bit 0 était à 1
Lecture/Polling	Commande (PS=0)	Registre de status / Attente du bit **BUSY** (LM occupé) à 0
Ecriture	Donnée (PS=1)	**8 bits poids fort** du terme limite de l'intégrale de l'erreur
Ecriture	Donnée (PS=1)	**8 bits poids faible** du terme limite de l'intégrale de l'erreur

Update Filter :

Lecture/Ecriture	Registre	Valeur
Lecture/Polling	Commande (PS=0)	Registre de status / Attente du bit **BUSY** (LM occupé) à 0
Ecriture	Commande (PS=0)	0x04 (**UDF**)

Pour tout renseignement il y a une hotline :

pcarlier@etu.utc.fr (06 75 84 21 81) (en ce qui me concerne mieux vaut téléphoner que d'envoyer un mail, ou alors il faut m'avertir par téléphone que j'ai un mail…)

gtrannoy@etu.utc.fr (06 71 19 22 40)